Operating System

Project #5

16. 11. 28

Project Plan

- 5 projects
  - Install Xv6
  - System call + scheduling
  - Virtual memory (stack growth + COW)
  - Thread-support
  - Concurrency

- Single-handed project

Syncronization on Xv6

- Support mutex & condition variable on xv6
- Define data structures for mutex and condition variable
- Implement new system calls
  - mutex_init()
  - mutex_lock()
  - mutex_unlock()
  - cond_init()
  - cond_wait()
  - cond_signal()
- Threads are blocked(sleep) on mutex_lock() and cond_wait()
- Consider thread priority(nice) when unblock waiting threads
  - The highest priority thread should be unblocked

Synchronization on Xv6 – mutex_init()

- Name

  mutex_init – initialize a mutex

- Synopsis

  int mutex_init(struct mutex_t *mutex)

- Description

  The mutex_init() initializes the mutex referenced by mutex. Upon successful initialization, the state of the mutex becomes initialized and unlocked.

- Return value

  If successful, the mutex_init() function returns 0. It returns -1 when the value specified by mutex is invalid, -2 when attepting to reinitialize an already initialized mutex, or -3 when the mutex cannot be initialized for other reasons.

Synchronization on Xv6 – mutex_lock()

- Name

  mutex_lock – lock a mutex

- Synopsis

  int mutex_lock(struct mutex_t *mutex)

- Description

  The mutex object referenced by mutex is locked by calling mutex_lock(). If the mutex is already locked, the calling thread blocks until the mutex becomes available.

- Return value

  If successful, the mutex_lock() returns 0. It returns -1 when the value specified by mutex is invalid, -2 when the mutex is not initialized, or -3 when the current thread already owns the mutex.

Synchronization on Xv6 – mutex_unlock()

- Name

  mutex_unlock – unlock a mutex

- Synopsis

  int mutex_unlock(struct mutex_t *mutex)

- Description

  The mutex_unlock() releases the mutex object referenced by mutex. If there are threads blocked on the mutex, thr highest priority thread waiting for the mutex should be unblocked and put on the list of ready thread.

- Return value

  If successful, the mutex_unlock() returns 0. It returns -1 when the value specified by mutex is invalid, -2 when the mutex is not initialized, or -3 when the current thread does not owns the mutex.

Synchronization on Xv6 – cond_init()

- Name

  cond_init – initialize a condition variable

- Synopsis

  int cond_init(struct cond_t *cond)

- Description

  The cond_inti() initializes the condition variable referenced by cond. Upon successful initialization, the state of the condition variable becomes initialized.

- Return value

  If successful, the cond_init() returns 0. It returns -1 when the value specified by cond is invalid, -2 when attempting to reinitialize an already initialized condition variable, or -3 when the condition variable cannot be initialized for other reasons.

Synchronization on Xv6 – cond_wait()

- Name

  cond_wait – initialize a condition variable

- Synopsis

  int cond_wait(struct cond_t *cond, struct mutex_t *mutex)

- Description

  The cond_wait() blocks on a condition variable. It should be called with mutex locked by the calling thread. It should release mutex and cause the calling thread to block on the condition variable cond. Upon successful return, the mutex should be locked and owned by the calling thread.

- Return value

  If successful, the cond_wait() returns 0. It returns -1 when the value specified by mutex or cond is invalid, -2 when the mutex or condition variable is not initialized, or -3 when the mutex was not owned by the current thread.

Synchronization on Xv6 – cond_signal()

- Name

  cond_signal – signal a condition

- Synopsis

  int cond_signal(struct cond_t *cond)

- Description

  The cond_signal() unblocks a threead blocked in the specidied condition variable cond. If more than one thread is blocked on the condition variable, the highest priority thread waiting for the condition variable should be unblocked and put on the list of ready threads.

- Return value

  If successful, the cond_signal() returns 0. It returns -1 when the value specified by cond is invalid, -2 when the condition variable is not initialized.

Synchronization on Xv6

- Our condition variable follows Mesa semantics(opposite, Hoare). cond_signal() places an unblocked thread on the ready queue, but the signaler continues inside the critical section.

- As in Project #4, the maximum number of threads per process os limited to 8 (including main thread). This means that the number of threads that are blocked on a mutex or a condition variable does not exceed 7.

Project #5 – Synchronization

- Implement mutex & condition variable in xv6
- Submit a tar.gz file
- Send email to T.A
  - [SSE3044]Project#5-YOURID-YOURNAME
    - ex) [SSE3044]Project#5-2016710580-leegyusun
  - Email address : lgs0409@naver.com
- Due date
  - 2016-12-11(Sun) PM 23:59
  - -10% per day (until 12/14)